Simulation Support for Fast and Accurate Large-Scale GPGPU & Accelerator Workloads

Vishnu Ramadas, Matthew Poremba, Bradford M. Beckmann, Matthew D. Sinclair
University of Wisconsin-Madison, AMD Research
vramadas@wisc.edu Matthew.Poremba@amd.com Brad.Beckmann@amd.com sinclair@cs.wisc.edu

I. Introduction

In recent years deep neural networks (DNNs) have emerged as an important application domain driving the requirements for future systems. As DNNs get more sophisticated, their compute requirements and the datasets they are trained on continue to grow at a fast rate. For example, Gholami showed that compute in Transformer networks [1] grew 750X over 2 years [2], while other work projects DNN compute and memory requirements to grow by 1.5X per year [3], [4]. Given their growing requirements and importance, heterogeneous systems often add machine learning (ML) specific features (e.g., TensorCores) to improve their efficiency. However, given ML's voracious rate of growth and size, there is a growing challenge in performing early-system exploration based on sound simulation methodology.

Traditionally, architectural simulators are used to perform early exploration for this type of research. However, detailed simulation of modern systems can take extremely long times in existing tools. Furthermore, prototyping optimizations at scale can also be challenging, especially for newly proposed accelerators. Although tools such as Accel-Sim [5], [6], Gemmini [7], MGPUSim [8], [9], and SCALE-Sim [10] enable some early experiments, they are limited in their ability to target a wide variety of accelerators and often focus on specific accelerators instead of system-wide behavior. Likewise, approaches like Path Forward [11] and Photon [12] reduce GPU simulation time by approximating the ML application behavior. However, these approaches focus on existing hardware – prototyping new optimizations on them would be challenging.

In comparison, gem5 [13], [14] has support for various CPUs, GPUs, and other important accelerators [15]–[18]. However, efficiently simulating large-scale workloads on gem5's cycle-level models requires prohibitively long times. Accordingly, we are enhancing gem5's support to make these workloads practical to run while retaining accuracy. This will enable users to use the same early exploration platform, gem5, for both homogeneous and heterogeneous systems. Specifically, we are extending gem5's existing simulation infrastructure to create a scalable simulation framework that models diverse system components with varying levels of detail. We will also develop novel mechanisms that identify the most important parts of the applications to simulate at high fidelity while reducing simulation time and add support for multi-chiplet and multi-node (e.g., multi-GPU and multi-accelerator) simulations to drive a flexible, adaptable simulation infrastructure that enables rapid prototyping of different optimized AI and ML algorithms and architectures while ensuring that the changes are representative of real, modern hardware and software.

II. Background

The gem5 simulator is a widely used, open-source, cycle-level computer system simulator. At its core, gem5 contains an event-driven simulation engine. On top of this simulation engine gem5 implements a large number of models for system components for CPUs (out-of-order designs, in-order designs, and others), GPUs (AMD and ARM models), accelerators [17], [19], various memories, on-chip interconnects, coherent caches, I/O devices, and many others. Moreover, gem5 provides two modes: Syscall Emulation (SE) and Full System (FS). SE mode simulates an application's user mode code in detail but emulates the OS instead of simulating it in detail. Conversely, FS mode simulates both the OS and user mode code in detail, allowing users to study the interaction between the OS and architecture.

Our recent work enhanced and updated gem5's GPU support [20] to add support for multi-chiplet setups [18] and ML workloads [16], [21] in gem5's SE mode. However, this support focuses on relatively smaller ML workloads such as DNNMark [22] and DeepBench [23] which call ML libraries directly (unlike high-level frameworks like PyTorch or TensorFlow). To improve on this, we recently released support for running ML models in gem5's FS mode in gem5 v23.1. As a result, users can now run PyTorch and TensorFlow workloads on CPU-GPU systems in gem5 using modern versions (e.g., v6) of AMD's open-source ROCm stack. Unfortunately, running ML models from these high-level frameworks in gem5 is extremely slow: it would take roughly 78 days of simulation time to reach the region of interest in massive modern ML workloads running in PyTorch or TensorFlow. Thus, significant work is needed to practically simulate these large ML workloads in gem5.

III. On-Going Work

Here we discuss our on-going efforts to improve gem5's support to practically run ML workloads and other large-scale workloads (e.g., from high performance computing). These optimizations, which we have open sourced to the public, mainline gem5 codebase, trade-off fidelity for less important portions of the workload for improved simulation time, without compromising correctness. Moreover, we are currently working on ways to further extend this to improve simulation speed without compromising accuracy.

Reduce Fidelity for Less Important Workload Portions with KVM: We extended gem5's KVM CPU support to speed up simulation for gem5 applications running on CPUs, GPUs, and other accelerators. We currently harness gem5's KVM CPU to simulate the CPU code of a GPGPU workload at native speed on the underlying machine. We simulate the workload kernels at high fidelity using gem5's GPU models while using the KVM CPU for everything else. However, even after doing so, if an application has several iterations or a large number of phases (e.g., GPU kernels or application phases) that require high fidelity simulation, further optimizations are required [5], [11], [12].

Checkpoint Save/Restore Support: We also added support for checkpointing applications and are adding support to allow gem5 users to create their own checkpoints and restore from them. For example, users could create a checkpoint for the state of the system immediately before a region of interest (ROI) executes or immediately before the first non-setup phase executes. The checkpointed state can then be restored during later simulations to skip over all instructions until the annotation and effectively begin simulating from the ROI onwards. This significantly improves simulation time by only simulating these less important parts (e.g., if a user does not care about reading inputs in [24]–[26]) once. Furthermore, for GPU phases the ROCm stack can also be modified to generate a checkpoint from hardware that contains all the required state information, and recent work has demonstrated that similar support exists for CPUs [27]. This checkpoint can then be used with gem5 to start a simulation directly from the ROI.

Fast Forward Through Less Important Phases: We propose to identify and fast forward through less important application phases (e.g., start-up GPU kernels). Here, like prior work, we observe that some code regions in a workload are more important to the application's overall behavior than others. By converting less important GPU kernels to CPU phases that can be executed natively via KVM or by executing them by passing through the host machine's GPU itself, we can simulate the workload much faster by focusing the high fidelity simulation on the most important application phases. Our initial prototype (using HIP-CPU [28] and simple GPGPU benchmarks from Rodinia [29], [30]) shows this approach is highly effective – gem5's simulation time is only 1.6×-3× slower than bare metal (versus at least 200× slower without these optimizations). Moving forward, we plan to leverage the fact that high level frameworks have multiple backends for kernels. Accordingly, applying this to large-scale ML workloads will make their simulation tractable – allowing relatively small ML workloads to complete very quickly and making it possible to simulate much larger workloads that currently have infeasibly long runtimes. However, significant challenges exist for generating and migrating the state needed to initiate cycle-level simulation after the fast forward phase.

Annotating Applications: Fast forwarding through less important phases requires a simple mechanism to identify which phases/GPU kernels are most important. To do this, we will initially manually annotate applications to identify which kernels require high fidelity. However, this may not be feasible for larger workloads. Thererfore, we will develop a novel profiling scheme that analyzes large-scale ML workloads on real machines and identifies which ROIs are most important to simulate in high fidelity – by passing this information into a scripting framework we will develop, we can automatically identify the most important regions without requiring manual annotation. By utilizing the aforementioned annotations for the ROIs, we can also make it easy for users to create checkpoints for ML workloads. Here, we propose to start with our prior work on SeqPoints [31], which identifies a subset of the computation in RNNs and Transformers that are representative of the larger ML training computation. Applying SeqPoints to gem5 will create a set of profiles and checkpoints representative of the larger workloads that can be run in gem5. Our analysis shows that these profiles are up to 345× smaller than the entire ML workloads, enabling significant savings for RNNs and Transformers. For other important ML workloads, we will either adopt state-of-the-art approaches (e.g., only simulating a single CNN iteration [32] or using clustering [5]) or perform a detailed analysis of the workloads to identify what subset of the workload must be simulated at high accuracy.

Turnkey Ease of Use: One of the barriers to entry with using simulators such as gem5 is the difficulty in getting the tools set up properly initially. Although recent work has helped improve this [16], [21], it does not work with our proposed work. We have begun developing detailed documentation for guiding users to setup their simulation environments. We also plan to include a new chapter in the learning gem5 book as well as example applications and uses in gem5-resources [21]. These resources will allow new users to simply start running PyTorch or TensorFlow applications without worrying about how to properly configure gem5, and give them the tools needed to create their checkpoints for additional applications.

IV. Methodology

We will evaluate our proposed work in the gem5 simulator, and plan to continue open-sourcing this work. Our main metrics for evaluation will be a) simulation time (wall clock time) compared to the current gem5 support for these applications and b) accuracy (i.e., how well the simulated support models the behavior of applications on a given system). Here, we will leverage our recent work on improving the accuracy of gem5's models [33], [34]. Directly quantitatively evaluating its efficacy compared to prior work (discussed in Section I) is challenging since each uses a different simulator or ecosystem. Moreover, our work evaluates the entire application, end-to-end, whereas prior work specifically analyzes the accelerator portion (e.g., the GPU kernels). However, where possible we plan to both use the same/similar applications as PKA [5], Photon [12], and Path Forward [11] to demonstrate how our findings agree and contrast.

Acknowledgments

This work is supported in by the Semiconductor Research Corporation (SRC) and National Science Foundation grant Frameworks-2311889.

References

[1] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," 2019.

[2] A. Gholami, "AI and Memory Wall," 2021.

[3] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families : Industrial Product," in *ACM/IEEE 48th Annual International Symposium on Computer Architecture*, ser. ISCA, 2021, pp. 57–70.

[4] N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten Lessons from Three Generations Shaped Google's TPUv4i," in *Proceedings of the 48th Annual International Symposium on Computer Architecture*, ser. ISCA, 2021.

[5] C. Avalos Baddouh, M. Khairy, R. N. Green, M. Payer, and T. G. Rogers, "Principal Kernel Analysis: A Tractable Methodology to Simulate Scaled GPU Workloads," in *54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO. New York, NY, USA: Association for Computing Machinery, 2021, p. 724–737. [Online]. Available: https://doi.org/10.1145/3466752.3480100

[6] M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, "Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ser. ISCA, 2020, pp. 473–486.

[7] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao, A. Ou, C. Schmidt, S. Steffl, J. Wright, I. Stoica, J. Ragan-Kelley, K. Asanovic, B. Nikolic, and Y. S. Shao, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 769–774.

[8] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2Sim: A simulation framework for CPU-GPU computing," in *Proceedings of the 21st international conference on Parallel Architectures and Compilation Techniques*, ser. PACT, 2012.

[9] Y. Sun, T. Baruah, S. A. Mojumder, S. Dong, X. Gong, S. Treadway, Y. Bao, S. Hance, C. McCardwell, V. Zhao, H. Barclay, A. K. Ziabari, Z. Chen, R. Ubal, J. L. Abellán, J. Kim, A. Joshi, and D. Kaeli, "MGPUSim: Enabling Multi-GPU Performance Modeling and Optimization," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 197–209. [Online]. Available: https://doi.org/10.1145/3307650.3322230

[10] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, 2020, pp. 58–68.

[11] Y. Li, Y. Sun, and A. Jog, "Path Forward Beyond Simulators: Fast and Accurate GPU Execution Time Prediction for DNN Workloads," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 380–394. [Online]. Available: https://doi.org/10.1145/3613424.3614277

[12] C. Liu, Y. Sun, and T. E. Carlson, "Photon: A Fine-Grained Sampled Simulation Methodology for GPU Workloads," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1227–1241. [Online]. Available: https://doi.org/10.1145/3613424.3623773

[13] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[14] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, S. Bharadwaj, G. Black, G. Bloom, B. R. Bruce, D. R. Carvalho, J. Castrillon, L. Chen, N. Derumigny, S. Diestelhorst, W. Elsasser, M. Fariborz, A. Farmahini-Farahani, P. Fotouhi, R. Gambord, J. Gandhi, D. Gope, T. Grass, B. Hanindhito, A. Hansson, S. Haria, A. Harris, T. Hayes, A. Herrera, M. Horsnell, S. A. R. Jafri, R. Jagtap, H. Jang, R. Jeyapaul, T. M. Jones, M. Jung, S. Kannoth, H. Khaleghzadeh, Y. Kodama, T. Krishna, T. Marinelli, C. Menard, A. Mondelli, T. Mück, O. Naji, K. Nathella, H. Nguyen, N. Nikoleris, L. E. Olson, M. Orr, B. Pham, P. Prieto, T. Reddy, A. Roelke, M. Samani, A. Sandberg, J. Setoain, B. Shingarov, M. D. Sinclair, T. Ta, R. Thakur, G. Travaglini, M. Upton, N. Vaish, I. Vougioukas, Z. Wang, N. Wehn, C. Weis, D. A. Wood, H. Yoon, and Éder F. Zulian, "The gem5 simulator: Version 20.0+," 2020.

[15] A. Gutierrez, B. M. Beckmann, A. Dutu, J. Gross, M. LeBeane, J. Kalamatianos, O. Kayiran, M. Poremba, B. Potter, S. Puthoor, M. D. Sinclair, M. Wyse, J. Yin, X. Zhang, A. Jain, and T. Rogers, "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," in *2018 IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA, Feb 2018, pp. 608–619.

[16] K. Roarty and M. D. Sinclair, "Modeling Modern GPU Applications in gem5," in *3rd gem5 Users' Workshop*, June 2020.

[17] S. Rogers, J. Slycord, M. Baharani, and H. Tabkhi, "gem5-SALAM: A System Architecture for LLVM-based Accelerator Modeling," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2020, pp. 471–482.

[18] B. W. Yogatama, M. D. Sinclair, and M. M. Swift, "Enabling Multi-GPU Support in gem5," in *3rd gem5 Users' Workshop*, June 2020.

[19] Y. S. Shao, S. L. Xi, V. Srinivasan, G.-Y. Wei, and D. Brooks, "Co-designing accelerators and SoC interfaces using gem5-Aladdin," in *49th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO, 2016, pp. 1–12.

[20] A. Gutierrez, B. M. Beckmann, A. Dutu, J. Gross, M. LeBeane, J. Kalamatianos, O. Kayiran, M. Poremba, B. Potter, S. Puthoor, M. D. Sinclair, M. Wyse, J. Yin, X. Zhang, A. Jain, and T. Rogers, "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," in *2018 IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA, Feb 2018, pp. 608–619.

[21] B. R. Bruce, A. Akram, H. Nguyen, K. Roarty, M. Samani, M. Fariborz, T. Reddy, M. D. Sinclair, and J. Lowe-Power, "Enabling Reproducible and Agile Full-System Simulation," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, 2021.

[22] S. Dong and D. Kaeli, "DNNMark: A Deep Neural Network Benchmark Suite for GPUs," in *Proceedings of the General Purpose GPUs*, ser. GPGPU. New York, NY, USA: ACM, 2017, pp. 63–72. [Online]. Available: http://doi.acm.org/10.1145/3038228.3038239

[23] S. Narang and G. Diamos, "An update to DeepBench with a focus on deep learning inference," https://svail.github.io/DeepBench-update/, 2017.

[24] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "MLPerf Inference Benchmark," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ser. ISCA, 2020, pp. 446–459.

[25] P. Mattson, C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. A. Patterson, H. Tang, G. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. M. Hazelwood, A. Hock, X. Huang, B. Jia, D. Kang, D. Kanter, N. Kumar, J. Liao, G. Ma, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, V. J. Reddi, T. Robie, T. S. John, C. Wu, L. Xu, C. Young, and M. Zaharia, "MLPerf Training Benchmark," *CoRR*, vol. abs/1910.01500, 2019. [Online]. Available: http://arxiv.org/abs/1910.01500

[26] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, and C.-J. Wu, "The Vision Behind MLPerf: Understanding AI Inference Performance," *IEEE Micro*, vol. 41, no. 3, pp. 10–18, 2021.

[27] B. Gottschall, S. Santana, and M. Jahre, "Balancing Accuracy and Evaluation Overhead in Simulation Point Selection," in *IEEE International Symposium on Workload Characterization*, ser. IISWC, October 2023.

[28] A. Voicu, "HIP CPU Runtime," https://github.com/ROCm-Developer-Tools/HIP-CPU, 2023.

[29] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *IEEE International Symposium on Workload Characterization*, ser. IISWC, Oct 2009, pp. 44–54.

[30] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, Liang Wang, and K. Skadron, "A Characterization of the Rodinia Benchmark Suite with Comparison to Contemporary CMP Workloads," in *IISWC*, 2010, pp. 1–11.

[31] S. Pati, S. Aga, M. D. Sinclair, and N. Jayasena, "SeqPoint: Identifying Representative Iterations of Sequence-based Neural Networks," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, August 2020.

[32] H. Zhu, M. Akrout, B. Zheng, A. Pelegris, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "TBD: Benchmarking and Analyzing Deep Neural Network Training," in *IEEE International Symposium on Workload Characterization*, ser. IISWC, October 2018.

[33] C. Jamieson, A. Chandrashekar, I. McDougall, and M. D. Sinclair, "GAP: gem5 GPU Accuracy Profiler," in *4th gem5 Users' Workshop*, June 2022.

[34] V. Ramadas, D. Kouchekinia, N. Osuji, and M. D. Sinclair, "Closing the Gap: Improving the Accuracy of gem5's GPU Models," in *5th gem5 Users' Workshop*, June 2023.